

Simulating Non Stationary Operators in Search Algorithms

Adrien Goëffon

LERIA, University of Angers (France)

Frédéric Lardeux

LERIA, University of Angers (France)

Frédéric Saubion

LERIA, University of Angers (France)

Abstract

In this paper, we propose a model for simulating search operators whose behaviour often changes continuously during the search. In these scenarios, the performance of the operators decreases when they are applied. This is motivated by the fact that operators for optimization problems are often roughly classified into exploitation operators and exploration operators. Our simulation model is used to compare the different performances of operator selection policies and clearly identify their ability to adapt to such specific operators behaviours. The experimental study provides interesting results on the respective behaviours of operator selection policies when faced to such non stationary search scenarios.

Keywords: Island Models, Adaptive Operator Selection

1. Introduction

Selecting the most suitable operators in a search algorithm when solving optimization problems is an active research area (Eiben et al., 2007; Lobo et al., 2007). Given an optimization problem, a search algorithm mainly consists in applying basic solving operators — heuristics — in order to explore and exploit the search space for retrieving solutions. The choice of the successive operators along the search process is often driven by means

September 8, 2014

of parameters. The improvement of the performance of the algorithm thus relies on an adequate setting of these parameters. An optimal setting may be achieved by an optimal operator selection policy. Unfortunately, according to the underlying intuitions provided by the No Free Lunch theorems for optimization (Wolpert and Macready, 1997), this optimal policy may strongly depend on the instances of the problems to be solved. Initial parameters setting can be achieved by automated tuning algorithms (Hutter et al., 2009; Nannen et al., 2008). Nevertheless, the values of the parameters may require more continuous control (Fialho, 2010) and should rather not be fixed during the whole search process. Adaptive operator selection is strongly related to reinforcement learning problems, and especially to multi-armed bandit problems (Fialho et al., 2008; Costa et al., 2008). Various methods for managing the famous *exploration vs. exploitation* balance in search heuristics have been investigated in the literature; see for instance (Maturana et al., 2009; Lobo et al., 2007; Thierens, 2005). The performance of adaptive selection policies depends on the characteristics of the problem’s search space, as well as on the specificities of the search operators. Therefore different families of practical problems have been handled, but also more abstract operators models in order to provide more general and comprehensive testing frameworks as in Thierens (2005) and Costa et al. (2008), taking into account changes in the operators behaviours.

Motivations

In this paper, we propose an alternative model for simulating search operators whose behaviour often change continuously during the search. In these scenarios, the performance of the operators decreases when they are applied. This is motivated by the fact that operators for optimization problems are often roughly classified into exploitation operators and exploration operators. Exploitation operators aim at focusing on the evaluation of the visited configurations of the search space in order to converge quickly to a local optimum. Exploration operators aim at diversifying the search trajectory by visiting sparse areas of the search space. Unfortunately, it is not possible to always exploit nor explore the search space. For instance, it is unlikely that an exploitation operator will always improve a configuration and find directly an optimal solution (except for simple problems). Therefore, decreasing performance may be observed along the search as well as changing behaviours of operators.

Contributions

Our simulation model is used to compare the different performances of operator selection policies and clearly identify their ability to adapt to such specific operators behaviours and can be used as a surrogate operator model when designing new adaptive search algorithms. Hence, the general description of operator based search algorithms may be helpful in this design process when the user has to precisely identify the components and performance criteria that are used in the adaptive process. The experimental study provides interesting results on the respective behaviours of operator selection policies when faced to such non stationary search scenarios. Considered as a multi-armed bandit problem, our model corresponds to a specific restless bandit problem that could be used to model different real applications as soon as the efficiency of a given action decreases according to successive frequent uses. For instance, such reinforcement learning techniques are used to schedule on-line advertisement display on web pages. Our model could be pertinent in this context since it may be clear that the relevance of an advertisement decreases if it is too much shown to the same user. Other cases of such decrease repeated actions may actually be observed in various application domains.

Organization of the paper

In section 2, we describe optimization algorithm that are based on applications of basic search operators. We also define the problem of designing the best possible operator selection policy and show its relationship with multi-armed bandit problems. Section 3 is dedicated to review different operator selection policies. Section 4 presents our model for simulating non stationary operators. Experiments are presented in section 5.

2. Operator Based Search Algorithms for Optimization Problems

In this section, we propose to precisely define the components of a search algorithm in the context of solving optimization problems, in order to be able to handle their behaviour. We focus on algorithms that use operators — also commonly called heuristics — in order to find solutions in the search space. We introduce the notion of policy, that defines how the operators should be scheduled in a search process, and some criteria in order to compare different policies and thus to characterize an optimal policy.

Definition 1 (Optimization Problem). An optimization problem is a pair (\mathcal{S}, f) where \mathcal{S} is a search space whose elements represent solutions (or con-

figurations) of the problem and $f : \mathcal{S} \rightarrow \mathbb{R}$ is an objective function. An optimal solution (for maximization problems) is an element $s^* \in \mathcal{S}$ such that $\forall s \in \mathcal{S}, f(s^*) \geq f(s)$.

Example 1. As a simple running example, we may consider the toy *One-Max* problem, where $\mathcal{S} = \{0, 1\}^n$, for a given size n . Given a solution $s = (s_1, \dots, s_n) \in \mathcal{S}$, s_i is thus the value of the i^{th} bit of s . The objective function is simply $f(s) = \sum_{i=1}^n s_i$.

A general solving algorithm may be abstracted according to the following components.

Definition 2 (Operator Based Algorithm (OBA)). Given an optimization problem (\mathcal{S}, f) , a solving algorithm is defined by a tuple $(Init, \Omega, \theta, \pi)$:

- an initialization function $Init : 2^{\mathcal{S}} \rightarrow 2^{\mathcal{S}}$,
- a set of operators $\Omega = \{o_1, \dots, o_n\}$, which are functions $o_i : 2^{\mathcal{S}} \rightarrow 2^{\mathcal{S}}$,
- a parameter vector $\theta \in \Theta$, where Θ is a parameter space¹,
- a search policy $\pi : \Theta \times I \rightarrow \Omega$. $\pi \in \Pi$, where Π is the set of search policies and $I = \{1, \dots, T\} \subseteq \mathbb{N}$ is a set of iterations of the algorithm.

Typically, the initialization function takes as input² the whole search space — or a part of it — and returns a random element or a set of random elements of this search space. From a practical point of view, initialization is often insured by a random generation of points of the search space or by a greedy algorithm that builds a suboptimal solution. Note that we do not want to distinguish the different solving paradigms that use different data structures for their search processes, e.g., tree-based search, neighborhood search, population-based search... We only pay attention to the fact that these algorithms use basic search operators (branching heuristics, constraint propagation, hill climbing, recombination...) that they apply on configurations of the search space. The search policy determines the operator to

¹According to the classification proposed in (Smit and G.Eiben, 2009), parameters can be numerical or symbolic.

²This input represents indeed the knowledge that is required by the initialization function.

apply at each iteration of the algorithm. Therefore the notion of iteration is here directly linked to the application of one operator in a steady state fashion; this general model can anyway be adapted for coarser granularities of iterations.

Example 2. Back to our running example, let us consider a simple evolutionary algorithm for the One-Max problem. The algorithm includes an initialization function such that $Init(S) = \mathcal{P}_0$, where \mathcal{P}_0 is an initial population, randomly generated. The set of operators may include flipping operators (e.g., 1-flip is an operator that flips one bit — position — of a configuration $s \in \mathcal{S}$) and crossover operators (e.g., uniform crossover). Note that here, operators are applied on a population and include a selection and an insertion process. For instance, let $\mathcal{P} = \{000, 111, 110\}$, $1\text{-flip}(\mathcal{P}) = \{010, 111, 110\}$. The parameter vector could be $\theta = (psize, \sigma_1, \dots, \sigma_n)$ where $psize$ is the size of the population \mathcal{P} , and σ_i is the probability of application of an operator o_i . The search policy may be $\pi(\theta, t) = o_i$ with a probability σ_i , that corresponds to a roulette wheel selection as described in Section 3. Note that, for the moment, we consider that θ remains constant over iterations.

The operational semantics of a search algorithm can be defined by means of its runs. We define $\overline{\mathcal{S}}$ (resp. $\overline{\Omega}$) as the set of sequences over $2^{\mathcal{S}}$ (resp. Ω).

Definition 3 (Run of an OBA). A run of length l of an OBA $A = (Init, \Omega, \theta, \pi)$ for an optimization problem $P = (\mathcal{S}, f)$ is defined as a pair $(\overline{s}, \overline{o}) \in \overline{\mathcal{S}} \times \overline{\Omega}$ where:

- $\overline{s} = s^{(0)}, s^{(1)}, \dots, s^{(l)}$ with $s^{(0)} = Init(\mathcal{S})$
- $\overline{o} = o^{(1)}, \dots, o^{(l)}$
- $\forall i \geq 1, o^{(i)} = \pi(\theta, t)$, with
- $\forall i \geq 1, s^{(i)} = o^{(i)}(s^{(i-1)})$

The set of all possible runs of length l is denoted $Run_l(A, P)$.

Given an OBA A and an optimization problem P , the operational semantics of the algorithm is thus the set $Run(A, P) = \bigcup_{l \geq 0} Run_l(A, P)$. The runs may have different lengths depending on the stop criteria, e.g. reaching a solution, exceeding a maximum number of operators applications... Of

course, if the algorithm is fully deterministic, this set is a singleton (i.e., a single run for a given problem).

Given a run $r \in \text{Run}(A, P)$, we suppose that there exists a performance function such that $\text{Perf}(r) \in \mathbb{R}$. A run r is considered better than a run r' if $\text{Perf}(r) \geq \text{Perf}(r')$. This performance function is also supposed to be extended to subsets of runs $R \subseteq \text{Run}(A, P)$. This performance function may vary according to the type of algorithm and the type of problem (satisfaction, optimisation...). We now define a notion of optimality for the performance of the algorithms that corresponds to classic tuning point of view (Hoos, 2012). Various methods have been proposed to this aim (Birattari et al., 2002; Smit and G.Eiben, 2009; Hutter et al., 2009).

Definition 4 (Optimal Tuning). Given an optimization problem P , an OBA $A = (\text{Init}, \Omega, \theta, \pi)$ is optimally tuned for a problem P iff $\forall \theta' \in \Theta$, $\text{Perf}(\text{Run}(\text{Init}, \Omega, \theta', \pi), P) \leq \text{Perf}(\text{Run}(\text{Init}, \Omega, \theta, \pi), P)$.

Note that we decide here to consider tuning only on parameters in θ and not on the possible policies, which could also be consider as structural parameters — components — of the algorithm.

Only algorithms with fixed parameters and policy have been considered in the previous definitions. Of course, we have to take into account adaptive algorithms in which parameters may change during the search. Note that, in order to simplify our presentation, we do not consider changing policies. As defined above, a policy uses parameters and is then intrinsically submitted to adaptive changes by means of these parameters. Therefore, we may generalize our definition of an OBA to an adaptive OBA,.

Definition 5 (Adaptive OBA). An adaptive operator based algorithm is a tuple $A = (\text{Init}, \Omega, \theta, \pi, K)$ where $(\text{Init}, \Omega, \theta, \pi)$ is an OBA and K is a control function $K : \Theta \times I \rightarrow \Theta$ that provides a new parameter vector with regards to the current iteration.

The notion of run and operational semantics defined above can be extended straightforwardly to adaptive OBAs. We may now turn to optimal control of adaptive OBAs.

Definition 6 (Optimal Control). Given an optimization problem $P = (\mathcal{S}, f)$, an adaptive OBA $(I, \Omega, \theta, \pi, K)$ is optimally controlled for a problem P iff $\forall K'$, $\text{Perf}(\text{Run}(\text{Init}, \Omega, \theta, \pi, K'), P) \leq \text{Perf}(\text{Run}(\text{Init}, \Omega, \theta, \pi, K), P)$.

2.1. Abstracting Operators

We focus now on the control of the algorithms through their search policies and control functions. Therefore, we consider the notion of adaptive control policy as a pair (π, K) , which associates a search policy and a parameter control function. Following the terminology used in the literature (Lobo et al., 2007), such a policy is called an adaptive operator selection (AOS) policy. Note that the case where K is the identity function corresponds to a non adaptive policy. Therefore, in the remaining of the paper, we consider OS policies, including adaptive policies and fixed policies cases.

Example 3. Back to the example, we may consider an adaptive roulette wheel instead of the fixed one as previously mentioned. The control function may thus update the probability according to the observed performances of the operators by means of the function K , which is known as probability matching (see Section 3.1.1).

Given an optimization problem P , our purpose is to generate a policy (π, K) that produces optimal runs (\bar{s}, \bar{o}) , which may be abstracted as optimal sequences of operator applications \bar{o} . It is clear that the impact of the operators depends on the elements of \bar{s} on which they are applied. We consider that each operator provides a gain when it is applied; this gain may be evaluated according to the performance function.

Definition 7 (Gain of an operator). We define a gain function $g : \Omega \times \bar{\mathcal{S}} \rightarrow \mathbb{R}$, for an operator o , as follows.

$$g(o, \bar{s}) = \text{Perf}((\bar{s} \oplus o(s^{(k)}), \bar{o} \oplus o)) - \text{Perf}((\bar{s}, \bar{o}))$$

where $\bar{s} = s^{(0)}, \dots, s^{(k)}$; $\bar{o} = o^{(1)}, \dots, o^{(k)}$; $\bar{s} \oplus o(s^{(k)}) = s^{(0)}, \dots, s^{(k)}, o(s^{(k)})$ and $\bar{o} \oplus o = o^{(1)}, \dots, o^{(k)}, o$.

Note that the gain of an operator is defined according to the variation of gain that it provides on the current run. The gain of a policy can be computed for one run (definition 8).

Definition 8 (Gain of an OS Policy). Given a problem P and an OBA $(\text{Init}, \Omega, \theta, \pi, K)$, the gain obtained by the OS policy is thus the sum

$$G((\pi, K), r) = \sum_{o^{(k)} \in \bar{o}} g(o^{(k)}, (s^{(0)}, \dots, s^{(k)}))$$

such that $r = (\bar{s}, \bar{o}) \in \text{Run}((\text{Init}, \Omega, \theta, \pi, K), P)$.

The operator selection policy problem can be defined as follows.

Definition 9 (OS Policy Problem). Given $P = (\mathcal{S}, f)$, the components $Init$, Ω and θ of an OBA, the operator selection policy problem consists in finding a policy

$$(\pi^*, K^*) = \operatorname{argmax}_{\pi \in \Pi, K \in \mathcal{K}} \sum_{r \in Run(A, P)} G((\pi, K), r)$$

with $A = (Init, \Omega, \theta, \pi, K)$

Of course since most of the time, the whole set $Run(A, P)$ cannot be extensively computed, the optimal policy has to be approximated on a subset of $Run(A, P)$. From a practical point of view, there is a connection between the notion of optimal control and the notion of optimal policy.

Searching for an optimal policy which chooses at each step operators that maximize the overall gain may be related to bandit problems (Costa et al., 2008; Fialho, 2010).

2.2. Multi-Armed Bandits

The initial stochastic multi-armed bandit (MAB) (Robbins, 1952; Bradt et al., 1956; Rodman, 1978) is formulated as follow. Given several possible actions — usually called arms according to the gambling machine analogy — that have different individual gains (or rewards), one has to select a sequence of actions that maximizes the total gain. Definition 10 proposes a more formal definition of this general problem.

Definition 10 (Stochastic MAB). Let us consider n independent arms. For each arm $i \in \{1, \dots, n\}$, we have:

- a set of possible states S_i ;
- a set of probabilities $Prob_i = \{\sigma_{j \rightarrow k}^i, j, k \in S_i\}$ such that $\sigma_{j \rightarrow k}^i$ is the probability of being in state k if the arm i is played³ from state j ;
- a set of gains $G_i = \{g_j^i, j \in S_i\}$ where g_j^i is the gain obtained when arm i is played from state j .

³We use the verb *play* according *gain* to the gambling analogy.

Given a stochastic MAB, the problem is to find a policy that maximizes over a finite⁴ horizon T , $\sum_{t=0}^T g_t \gamma^t$, where g_t is the expected gain of the policy at time t and $\gamma \in [0, 1]$ is a discount factor.

Four features can be identified to characterize a MAB problem (Mahajan and Teneketzis, 2008):

1. only one arm is played at each time;
2. states of unplayed arms do not change;
3. arms are independent;
4. arms that are not played do not contribute any gain.

Many variants of the initial stochastic MAB have been studied in the literature. In this paper we focus on the restless MAB, first introduced in (Whittle, 1988). In this formulation, the gains of the arms change over time, while they are supposed to be fixed — but of course unknown — in the initial stochastic MAB formulation. In fact, restless bandits may be defined as in Definition 10 except that, when an arm is not played, its state may change, which corresponds to a relaxation of Feature 2. Hence, restless MABs involve two kinds of probabilities in $Prob_i$, namely $\sigma_{j \rightarrow k}^i$, which represents the probability of being in state k if the arm i is played, and $\tilde{\sigma}_{j \rightarrow k}^i$, that is the probability of being in state k if the arm i is not played.

3. Operator Selection Policies

In this sections we first explain how operator selection in operator based algorithms can be directly related to the choice of the most suitable sequence of actions in the context of multi-armed bandit problems. We review then different possible selection policies that can be used to achieved an optimal schedule of the operators.

3.1. Operator Selection in Operator Based Algorithms

Let us consider an OBA $A = (I, \Omega, \theta, \pi, K)$. Here, we are not interested in the initialization function I . Let $\Omega = \{o_1, \dots, o_n\}$ be the set of n operators.

⁴Note that we restrict the problem to finite horizon MAB. The most general problem is often presented over infinite horizon.

We have to define the control policy (π, K) which selects an operator at each iteration of the algorithm in order to build a run (\bar{s}, \bar{o}) . We review here different policies and we distinguish between policies based on probabilities of application of the operators and policies based on upper confidence bounds.

The gain of an operator (see Definition 7) is generally specific to the problem, since it uses the notion of performance of a run. In order to have a more general approach, a general notion of utility⁵, which reflects the successive gains obtained by the operators, can be introduced.

Considering a run (\bar{s}, \bar{o}) , such that $\bar{s} = s^{(0)}, \dots, s^{(n)}$ and $\bar{o} = o^{(1)}, \dots, o^{(n)}$, an utility $u_i^{(t)}$ is associated to each operator $i \in \{1..n\}$ for any iteration $t \in \{1..n\}$. This utility has to be re-evaluated at each time, classically using a formula $u_i^{(t)} = (1-\alpha)u_i^{(t-1)} + \alpha.g(o_i, s^{(0)}, \dots, s^{(t-1)})$, with $u_i^{(0)} = 0$. This utility uses the gain associated to the application of operator i (which corresponds thus to the immediate utility) and α which is a coefficient that controls the balance between past and immediate utilities, as in classic reinforcement learning techniques (Sutton and Barto, 1998). If an operator is not selected at iteration t , its gain is 0 for this iteration.

3.1.1. Policies based on probabilities of application

In this context, given the set of operators $\Omega = \{o_1, \dots, o_n\}$, we use the parameter vector θ to associate a probability of selecting the operator, $\theta = (\sigma_1, \dots, \sigma_n)$ such that $\sum_{i=1}^n \sigma_i = 1$. The search policy π is then a roulette selection wheel that selects each operator o_i according to its probability of selection σ_i . Different operator selection policies have been proposed in the literature (Lobo et al., 2007; Hamadi et al., 2012); we review here some of the most used of them.

- **Fixed Roulette Wheel**

A first possibility consists in keeping θ fixed during the run, i.e. $\forall t \in I, K(\theta, t) = \theta$. Note that these values can be determined by an automated tuning process (Hoos, 2012).

- **Adaptive Roulette Wheel**

Contrary to a static tuning of the operator application rates, adaptive operator selection consists in selecting the next operator to apply at

⁵Note that we use the term utility here, which should be clearly related to the notion of action value in reinforcement learning (Sutton and Barto, 1998).

iteration $t + 1$ by adapting the selection probability during the search. In this case, we have $\theta^{(t)} = (\sigma_1^{(t)}, \dots, \sigma_n^{(t)})$. The control function $K : \Theta \times I \rightarrow \Theta$ is defined as $K(\theta^{(t)}, t + 1) = \theta^{(t+1)}$. Defining K consists in defining the probabilities $\sigma_i^{(t+1)}$ with regards to the evolution of the operator's utilities.

A classic mechanism is the probability matching selection rule:

$$\sigma_i^{(t+1)} = p_{\min} + (1 - n \cdot p_{\min}) \frac{u_i^{(t+1)}}{\sum_{k=1}^n u_k^{(t+1)}}$$

where a non negative value p_{\min} insures a non zero selection probability for all operators. Note that, in order to insure a coherent behaviour, p_{\min} should be in the interval $[0, \frac{1}{n}]$.

- **Adaptive Pursuit**

An alternative proportional selection rule has been proposed in Thierens (2005), called adaptive pursuit (AP), that distinguishes the best current operator from the others:

$$\begin{cases} \sigma_{i^*}^{(t+1)} = \sigma_{i^*}^{(t)} + \beta(p_{\max} - \sigma_{i^*}^{(t)}) \\ \sigma_i^{(t+1)} = \sigma_i^{(t)} + \beta(p_{\min} - \sigma_i^{(t)}) \end{cases}$$

where $i^* \in \operatorname{argmax}_{i \in \{1, \dots, n\}} u_i^{(t+1)}$, $p_{\max} = 1 - (n - 1)p_{\min}$ and β is a parameter to adjust balance of this winner-take-all strategy.

3.1.2. Policies based on upper confidence bounds

Optimal strategies have been initially proposed by Feldman (1962) and Gittins (1979) for the multi-armed bandit problem. Later, Auer (2002) proposed to use this problem to manage the compromise between exploration and exploitation in optimization algorithms. The following policies consists in computing an upper confidence bound of the expected gain and to select thus the most promising arm.

- **UCB (upper confidence bound)**

The UCB1 criterion (Auer et al., 2002) is defined as:

$$\forall o_i \in \Omega, UCB1(o_i, t) = u_i^{(t)} + \sqrt{\frac{2 \log(\sum_{1 \leq k \leq n} nb_k^{(t)})}{nb_i^{(t)}}}$$

where $nb_i^{(t)}$ denotes the number of times operator o_i has been applied. Note that this formula is defined for gains that should be normalized between 0 and 1. The left term of the formula uses the successive utilities that are obtained by the arms in order to focus on the best arm, while the right term aims at providing the opportunity to be selected for less used arms. This formula attempts thus to achieve a compromise between exploitation and exploration.

Therefore, we may define the control policy (π, K) for a given iteration t as:

$$\begin{cases} \pi(\theta^{(t)}, t) = \underset{i \in \{1, \dots, n\}}{\operatorname{argmax}} \theta_i^{(t)} \\ K(\theta^{(t)}, t) = \theta^{(t+1)} = (UCB1(o_1, t+1), \dots, UCB1(o_n, t+1)) \end{cases}$$

Here no parameter is required. Note that UCB has originally been designed for fixed gain distributions. Since the gain of the operators is likely to change along the search, UCB has been extended to dynamic multi-armed bandit has to be considered.

- **DMAB (Dynamic MAB algorithm based on UCB)**

UCB has been revisited in Costa et al. (2008). A standard test — known as Page Hinkley (Hinkley, 1970) — for the change hypothesis is used. We may add a parameter in θ which indicates if the process has to be restarted. In this case the control function K uses the Page Hinkley test to detect statistical changes in the successive utilities of the operators and may re-initialize the values of the operators utilities. Moreover, it can be useful to add a scaling factor to the right term of the UCB1 formula in order to take into account the value range for utilities. The test is parametrized by γ that controls its sensitivity and δ that manages its robustness. We refer the reader to Fialho et al. (2010) for more details.

3.2. Policies based on Transition Matrix

Based on previous work on island models for operators selection (Candan et al., 2012), we present here a selection policy based on a probabilistic transition matrix. The underlying motivation is not only to detect the best possible operators but also possible relationships between operators.

Given an OBA A , we define a matrix M of size $|\Omega| \times |\Omega|$. $M(o_i, o_j)$ represents the probability of applying operator o_j after having applied o_i . According to our previous notations, M corresponds to the θ parameter of the selection policy (introducing a two-dimensional parameter structure).

From an operational point of view, contrary to previous policies, this policy uses simultaneously several possible runs of the OBA in order to acquire its knowledge. Let us consider a set $P^{(t)}$ of *psize* runs of length $t + 1$ produced by the algorithm A at iteration t (remind that runs are numbered from index 0 in definition 3). *psize* is a parameter of this policy. For each operator $o_i \in \Omega$, let $P_i^{(t)}$ be the set of runs $(s^{(0)} \dots s^{(t)}, o^{(1)} \dots o^{(t)}) \in P^{(t)}$ where $o^{(t)} = o_i$. Hence, this set is the set of all runs whose last applied operator is o_i , and we have $P^{(t)} = \bigcup_{o \in \Omega} P_o^{(t)}$.

The control policy (π, K) can thus be defined from M . Firstly, the policy π is defined by a roulette selection whose probabilities are given by M . Secondly, the update of M is performed by the control function K as:

$$\begin{aligned} K(M^{(t)}(i, k), t) &= M^{(t+1)}(i, k) \\ &= (1 - \beta)(\alpha M^{(t)}(i, k) + (1 - \alpha)R_i^{(t)}(k)) + \beta N^{(t)}(k) \end{aligned}$$

where $N^{(t)}$ is a stochastic vector such that $\|N^{(t)}\| = 1$ and $R_i^{(t)}$ is the reward vector that is computed by using the utility of the operators that have been used after applying i . More precisely:

$$R_i^{(t)}(k) = \begin{cases} \frac{1}{|B|} & \text{if } k \in B, \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{with } B = \operatorname{argmax}_{o_k \in \Omega} \left(\max_{\{\bar{s} \in \bar{\mathcal{S}} \mid \exists r \in P_k^{(t)}, r = (\bar{s}, o^{(1)} \dots o^{(t-1)} = o_i, o^{(t)} = o_k)\}} g(o_k, \bar{s}) \right)$$

B is the set of the best operators that have been applied after an operator o_i , i.e. that have provided the best gain. We could also update $R_i^{(t)}(k)$ by using the mean of the improvements. Remark also that this policy does not use the utility which is indeed computed within the update process.

The parameter α represents the importance of the knowledge accumulated (inertia or exploitation) and β is the amount of noise, which is necessary to explore alternative possibilities. The influence of these parameters has been studied in Candan et al. (2012).

This approach can be related to reinforcement techniques for MDP (Markov Decision Processes) (Sutton and Barto, 1998). Nevertheless island models use several populations in order to learn simultaneously from several sequences of operators.

4. Modelling Scenarios for Gain Functions

In practice, the gain function g may be difficult to compute and is very specific to the problem at hand. Therefore we may approximate such a gain function by using distributions. Comparisons of operator selection policies would be then easier and faster. As previously defined, we use a gain function $g(o, t)$ which represents an estimation of the gain of an operator o if it is applied at iteration t . Several scenarios for modelling g can be envisioned.

In Definition 7, the gain associated to an operator is defined according to the performance improvements that it provides during a given run of the algorithm. Remind that, for an optimization problem $P = (f, \mathcal{S})$, the effect of operators can be defined according to the evaluation function f . For instance, given an element $s \in \mathcal{S}$, the variation of evaluation f induced by an operator o is given as $f(o(s)) - f(s)$. This function can be extended to $\overline{\mathcal{S}}$. Of course different performance criteria could be taken into account.

4.1. Fixed Gains

For each operator o , the gain g is defined independently from iteration t by a binomial probability distribution (p_o, g_o) , i.e., $\forall i \in I, \Pr[g(o, i) = g_o] = p_o$ and $\Pr[g(o, i) = 0] = (1 - p_o)$. Such distributions have been proposed in Costa et al. (2008) for studying operator selection policies.

In this context, if the values (p_o, g_o) are fixed during the whole run of the algorithm, then determining the best policy just consists in finding the operator that has the greatest expected gain $p_o \cdot g_o$. This is indeed a basic stochastic multi-armed bandit problem (Definition 10), where operators correspond to arms with only one state. Of course, it is unlikely that, in real optimization problems, the effect of an operator remains unaltered in a whole run. Therefore it would be more realistic and interesting to consider gain functions that may evolve during the run.

4.2. Epoch Based Gains

In Thierens (2005), uniform distributions are associated to each operator in different overlapping intervals of values. The distributions are fixed during a given number of iterations — called epoch — and then are re-assigned

according to a permutation. The gain of the operator is thus non stationary during the algorithm's execution and the AOS has to discover the best operator to apply at each epoch. Such scenarios have been studied using various techniques including adaptive pursuit (Thierens, 2007), dynamic UCB (Fialho et al., 2008) and genetic algorithms (Koulouriotis and Xanthopoulos, 2008).

4.3. Non stationary gains with sliding time windows

In this section we define a new scenario in order to model operators whose behaviour changes more continuously during the solving process. We want to consider more continuous changes in the gain distributions. The idea is to provide a model where the gain of an operator decreases proportionally to its use. In such a model, the AOS policy must not detect the best operator during an epoch but rather identify suitable sequences of operators.

Within a run (\bar{s}, \bar{o}) of length n of an OBA, such that $\bar{o} = o_1, \dots, o_n$, we compute the gain of operator o at iteration t thanks to a gain function $g_{wsize}(o, t)$, defined as:

$$g_{wsize}(o, t) = g_o \cdot \left(1 - \frac{Occ_{wsize}(\bar{s}, o, t)}{wsize} \right)$$

g_o is a fixed maximal gain of operator o , and $Occ_{wsize}(\bar{s}, o, t)$ is the number of applications of operator o during the last $wsize$ iterations. More formally, $Occ_{wsize}(\bar{s}, o, t) = |\{i \in 1..|\bar{w}|, \omega_i = o\}|$, where \bar{w} is the subsequence $o_{t-wsize}, \dots, o_{t-1}$ that records the $wsize$ last applied operators. $wsize$ is a fixed parameter of the scenario.

This scenario can be formalized as a restless MAB problem:

- An arm i is associated to each operator $o_i \in \Omega$.
- For each operator $o_i \in \Omega$, we define a set of states $S_i = \{0..(2^{(wsize+1)} - 1)\}$, such that each state represents the previous use of the arm by a binary number. For instance, for a window of size $wsize = 4$, the state 1001 means that the arm has been played at iteration $t - 1$ and $t - 4$ only.
- The set of transition probabilities $Prob_i$ between states can be defined as:

$$\begin{aligned}
1. \quad \sigma_{j \rightarrow k}^i &= \begin{cases} 1 & \text{if } k = Lshift(j) + 1 \\ 0 & \text{otherwise} \end{cases} \\
2. \quad \tilde{\sigma}_{j \rightarrow k}^i &= \begin{cases} 1 & \text{if } k = Lshift(j) \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

$Lshift(j)$ = denotes the logical left shift of a binary number of fixed size $wsiz$ e. Note that we need $\tilde{\sigma}$ probabilities since we are modelling a restless MAB problem (see definition 10 and section 2.2).

- When an arm i is played from state j , the reward can be straightforwardly defined as $r_j^i = g_o \cdot (1 - \frac{\#_1(j)}{wsiz})$, where $\#_1(j)$ is the number of bits being equal to 1 in state j .

This restless bandit problem involves a two states transition matrix whose size is in $O((n \cdot 2^{wsiz})^2)$. Of course in practice, one has to memorize the $wsiz$ previous applications of operators.

4.4. Non Stationary Binary Scenarios

In optimization algorithms, operators are often classified intuitively in two classes : diversification operators and intensification operators, whose behaviours should be understood as orthogonal. In order to better understand such scenarios we consider here two types of operators, according to the previous notations: $(1, 1)$ and $(1, 0)$, that we denote here respectively $\mathbb{1}$ and $\mathbb{0}$. The $\mathbb{1}$ operators theoretically always gain 1 but their efficiency will decrease proportionally to their use according to our sliding window. $\mathbb{0}$ always gain 0. We choose here probability of 1 for all operators in order to avoid probability side effects in our analysis. We also use gain values whose range is in $[0, 1]$.

An instance of a non stationary binary scenario can be fully defined by a triple $(N_{op}, N_1, wsiz)$ where N_{op} is the number of operators, N_1 is the number of $\mathbb{1}$ operators and $wsiz$ is the length of the window used for computing the decay of the operator's gains (see section 3).

It can be shown that, independently from the size of the windows, the score obtained by a uniform choice of the operators is constant:

Property 1. Given an instance $(N_{op}, N_1, wsiz)$, the expectation of gain for a $\mathbb{1}$ operator is:

$$\forall t \in I, \mathbb{E}[g(\mathbb{1}, t)] = 1 - \frac{1}{N_{op}}$$

Indeed, $\Pr[Occ_{size}(\bar{s}, o, t) = k] = \left(\frac{1}{N_{op}}\right)^k \left(1 - \frac{1}{N_{op}}\right)^{size-k} \binom{size}{k}$, with a corresponding gain of $\left(1 - \frac{k}{size}\right) \cdot p \cdot r$ for a (p, r) operator, i.e. $\left(1 - \frac{k}{size}\right)$ for a $\mathbb{1}$ operator. Considering that $Occ_{size}(\bar{s}, o, t)$ is a binomially distributed random variable of parameters $size$ and $\frac{1}{N_{op}}$, then $\mathbb{E}[Occ_{size}(\bar{s}, o, t)] = \frac{size}{N_{op}}$, and $\mathbb{E}[g(\mathbb{1}, t)] = \mathbb{E}\left[1 - \frac{Occ_{size}(\bar{s}, o, t)}{size}\right] = 1 - \frac{1}{N_{op}}$.

More generally, we have then the following property:

Property 2. Given an instance $J \equiv (N_{op}, N_1, size)$, the gain expectation per iteration is equal to $\frac{N_1}{N_{op}} \left(1 - \frac{1}{N_{op}}\right)$.

Therefore the behaviour of a uniform selection of the operators is independent from $size$, for a given N_{op} . The total expected gain for N_1 operators can easily be computed according to the number of allowed iterations. A uniform choice policy may thus serve as baseline for other selection processes (see Section 5).

We may suppose that the $\mathbb{1}$ operators are numbered $1..N_1$ and \mathbb{O} are numbered $N_1 + 1..N_{op}$. The optimal policy problem can be formulated as a discrete constraint optimization problem.

Definition 11 (Optimal Policy for Binary Non Stationary Scenarios).

Given a non stationary binary scenario $(N_{op}, N_1, size)$ and an horizon T (the number of allowed iterations) we define

- the set of variables $\mathcal{X} = \{x_1, \dots, x_T\}$, such that x_i is the operator applied at iteration i (decision variables)
- $\forall i \in \{1, \dots, T\}, x_i \in \{1, \dots, N_{op}\}$ (domains of the decision variables)
- $\forall j \in \{1, \dots, N_1\}, g_j = \frac{1}{size} \sum_{k=1}^{T-size} \left(\sum_{i \in k..k+size | x_i=j} i \right)$ (gains for all operators whose value is 1)
- the objective function is $\max \sum_{j \in \{1..N_1\}} g_j$

Due to the size of the induced search space, this problem cannot be solved practically for large values of T . Nevertheless, we may restrict this problem to a limited window and compute suboptimal policies. We consider a restricted model. Note that, since we are only interested in maximizing the gain, the \mathbb{O} operators are all equivalent and we may consider only one \mathbb{O} operator, i.e., $N_{op} = N_1 + 1$ or $N_{op} = N_1$ if no \mathbb{O} operator is considered. We consider here a circular scenario of length Sc on which the total gain is computed in order to simulate the successive repetitions of this scenario. Since we want to be able to compute circularly the total gain induced by such a scenario, we have to consider a computation sequence *comps* that is both a multiple of the length of the windows *wsiz*e and of the length of the scenario Sc (i.e., the least common multiple *lcm*).

Definition 12 (Suboptimal Policy for Binary NS Scenarios). Given a non stationary binary scenario $(N_{op}, N_1, wsiz$ e) and scenario length Sc (the number of allowed iterations of operators in the scenario) we define

- $comps = lcm(wsiz$ e, Sc) (the length of the computation sequence)
- $nbseq = comps \text{ div } Sc$ (the number of sequence of length Sc in the total computation windows)
- the set of variables $\mathcal{X} = \{x_1, \dots, x_{comps}\}$, such that x_i is the operator applied at iteration i (decision variables)
- $\forall i \in \{1, \dots, comps\}, x_i \in \{1, \dots, N_{op}\}$ (domains of the decision variables)
- $\forall i \in \{1, \dots, comp\}, k \in \{1, \dots, nbseq-1\}, (x_i = x_{i+(k*Sc)})$ (the scenario is repeated $nbseq$ times in the computation sequence)
- $\forall i \in \{(wsiz$ e + 1), $\dots, comps\}$ such that $x_i \in \{1, \dots, N_1\}$,
 $g_i = \frac{1}{wsiz$ e} \sum_{\{k \in (i-wsize).. $i | x_k = x_i\}}$ k (standard gains for operators in the computation sequence)
- $\forall i \in \{1, \dots, wsiz$ e\} such that $x_i \in \{1, \dots, N_1\}$,
 $g_i = \frac{1}{wsiz$ e} \left(\sum_{\{k \in 1..i | x_k = x_i\}} k + \sum_{\{k \in (comps-wsize+i).. $comps | x_k = x_i\}} k \right)$ (circular case)
- the objective function is $\max \sum_{j \in \{1..N_1\}} g_j$

	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.500	0.333	0.500	0.400	0.500	0.428	0.500	0.333	0.500	0.454	0.500	0.461	0.500	0.400
2	0.250	0.333	0.375	0.300	0.333	0.357	0.375	0.333	0.300	0.363	0.375	0.346	0.357	0.333
3	0.333	0.222	0.333	0.333	0.333	0.286	0.333	0.333	0.333	0.333	0.333	0.333	0.333	0.333
4	0.250	0.250	0.250	0.300	0.333	0.321	0.313	0.278	0.300	0.318	0.333	0.326	0.321	0.300
5	0.300	0.267	0.300	0.240	0.300	0.314	0.325	0.311	0.300	0.273	0.300	0.308	0.314	0.320
6	0.250	0.222	0.292	0.267	0.250	0.286	0.313	0.315	0.317	0.303	0.292	0.269	0.286	0.300
7	0.286	0.238	0.286	0.257	0.286	0.035	0.286	0.302	0.314	0.312	0.310	0.297	0.286	0.267
8	0.125	0.250	0.063	0.275	0.135	0.268	0.031	0.278	0.150	0.307	0.078	0.308	0.152	0.292

Table 1: Expected gain per iteration w.r.t. window size (*wsiz*e) on lines and computation sequences (*comps*) on column for $N_{op} = 2$ and $N_1 = 1$

This model can be used to compute sub-optimal policies. We have used the Minizinc (NICTA, 2014) constraint modelling and solving framework in order to compute such policies. Table 1 shows the expected gain per iteration for different scenario window sizes, considering one **0** operator and one **1** operator.

5. Experimental Results

In this section, we study the behaviour of different operator selection policies on binary non stationary scenarios. Note that, as mentioned above, the behaviour of these policies for fixed gain operators or epoch based scenarios has already been studied and will not be considered here.

5.1. Experimental Settings

The following notations are used for the different policies :

- *GR* is a basic greedy selection policy that always selects the operator with the current maximal utility. This policy is used without any learning stage, which cannot be efficient here since the gains of the operators continuously change.
- *EGR* is an ϵ -greedy policy that selects greedily the best operator according to its current utility, but uses an exploration coefficient ϵ . Therefore at each iteration the operator with the current maximal utility is selected with probability $1 - \epsilon$ and a randomly selected operator is used with probability ϵ . Note that *GR* and *EGR* are basic reinforcement

learning strategies. Despite their poor results, these policies just serve here as baseline to highlight that a simple operator policy cannot be efficient for our scenarios.

- *OR* is a *myopic* oracle that is aware of the gains of the operators and the last applied operators in order to select the next operator with the best expected value according to the variation of total gain . Note that this oracle is different from the suboptimal policy that can be computed using Definition 12.

This oracle does not compute the optimal solution for a given number of iterations since it does not take into account future operators applications that are used to compute gains. In particular, compare to the optimal and suboptimal policies described in section 4.4, *OR* does not fully make use of the size of the windows *wsiz*e.

- *U* is a uniform selection choice rule. It correspond to a fixed roulette with equal probabilities.
- *ARW* is an adaptive probability matching selection rule (adaptive roulette wheel) using a minimal probability p_{\min} , as defined in Section 3.1.1.
- *AP* is the adaptive pursuit method with parameters p_{\min} and β (See section 3.1.1).
- *UCB* uses the upper confidence bound UCB1 as described in Section 3.1.2.
- *DMAB* is the dynamic selection based on UCB1 with the Page Hinkley test. This method requires several parameters for the test (see description in Section 3.1.2).
- *IM* is the island model implementation of the policy based on transition matrix as described in Section 3.2. We do not detail the algorithmic implementation here and refer the reader to Candan et al. (2012) for more details.

We first set the experimental conditions:

- Each policy is run 20 times on each instance. Since island models use 80 individuals, the mean value (MeanV) corresponds to the average score of the 20 best individuals obtained at each run. In order to achieve fair comparisons, other policies have been run 20×80 times and the 20 best scores have been extracted from each sequence of 80 runs. Note that the value 80 has been chosen here since we use a scenario with 8 operators and we thus use sub-populations of size 10 for each operators.
- Parameters

Method	Parameters	Range Value	Tuned values
GR	-	-	-
EGR	ϵ	$[0, 1]$	0.05
OR	-	-	-
U	-	-	-
ARW	p_{\min}	$[0, 1]$	0.05
IM	α	$[0, 1]$	0.8
	β	$[0, 1]$	0.01
	$nbind$	\mathbb{N}	80
AP	β	$[0, 1]$	0.7
	p_{\min}	$[0, 1]$	0.1
UCB	-	-	-
DMAB	γ	$[0, \infty]$	0
	δ	$[0, \infty]$	0

- The policies have all been implemented using Scilab (Scilab Enterprise, 2014). Experiments have been run on a desktop computer with Intel Core i5 CPU, 2.6 GHz, 4 Go RAM. Parameters have been tuned using a principled approach inspired by F-Race (Birattari et al., 2002). Note that here there are indeed few parameters for these policies.

5.2. Results

The following tables 2 and 3 compare different methods for windows sizes $wsize$ varying from 1 to 8, using binary non stationary scenarios with $N_{op} = 8$ and N_1 varying from 1 to 8, as defined in Section 4.4. We report average scores of the best 20 runs and their standard deviation.

In order to highlight the respective advantages and drawbacks of the different selection policies, Figure 1 propose a graphical view of some of these results.

wsiz = 1																
N ₁	1		2		3		4		5		6		7		8	
OR	476.10	1.66	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00
IM	465.40	1.96	955.40	6.72	966.60	3.37	973.90	2.08	982.60	0.97	988.80	1.03	994.30	1.34	998.50	0.53
GR	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
EGR	57.30	2.00	227.70	54.86	245.90	40.95	278.60	32.64	318.00	34.66	340.20	28.90	355.60	24.11	365.80	26.68
U	129.80	3.52	245.60	4.93	356.90	3.38	472.10	4.84	584.30	8.15	691.90	5.26	794.10	3.41	901.80	4.83
UCB	86.00	0.00	526.70	129.23	909.10	96.23	958.30	0.67	969.90	0.32	977.00	0.00	984.00	0.00	991.00	0.00
ARW	249.30	2.41	481.20	7.54	594.10	9.30	665.30	3.68	736.00	4.40	793.50	6.92	844.70	4.76	895.60	2.99
AP	230.50	4.84	330.00	3.65	426.20	2.20	516.90	3.81	611.30	6.15	700.70	5.36	785.90	7.31	876.90	5.24
MAB	86.00	0.00	502.40	83.03	922.10	49.93	958.70	0.48	969.90	0.32	977.00	0.00	984.00	0.00	991.00	0.00
wsiz = 2																
N ₁	1		2		3		4		5		6		7		8	
OR	324.15	0.91	678.10	1.54	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00
IM	91.95	3.94	507.70	2.34	956.65	4.06	966.35	3.56	973.40	3.50	979.10	2.31	983.05	1.19	987.15	1.27
GR	1.50	0.00	1.50	0.00	1.50	0.00	1.50	0.00	1.50	0.00	1.50	0.00	1.50	0.00	1.50	0.00
EGR	57.30	3.23	303.20	59.41	342.70	62.10	365.90	26.24	398.60	38.25	448.10	60.37	464.75	47.86	472.45	39.13
U	129.95	2.23	244.15	4.81	360.50	5.31	468.95	4.42	580.00	6.84	684.50	4.55	788.35	3.27	892.75	2.60
UCB	116.00	0.00	676.45	0.37	700.85	0.85	944.10	21.13	971.65	0.24	979.25	0.26	986.50	0.00	994.00	0.00
ARW	245.45	3.37	473.95	2.25	586.50	2.40	661.70	5.14	730.05	4.07	785.75	6.17	838.20	4.09	888.45	2.59
AP	228.35	4.08	331.30	4.53	429.05	4.96	520.45	7.51	612.10	4.46	704.55	3.32	787.40	4.40	874.15	2.90
MAB	116.00	0.00	676.40	0.21	702.30	1.46	948.45	6.96	971.70	0.48	979.00	0.00	986.50	0.00	994.00	0.00
wsiz = 3																
N ₁	1		2		3		4		5		6		7		8	
OR	245.77	0.35	667.33	0.00	841.07	0.84	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00
IM	80.83	2.46	643.63	1.61	673.50	2.71	960.90	4.27	966.73	2.81	972.53	3.18	975.73	3.01	978.60	2.30
GR	2.00	0.00	2.00	0.00	2.00	0.00	2.00	0.00	2.00	0.00	2.00	0.00	2.00	0.00	2.00	0.00
EGR	59.63	2.25	314.03	49.56	408.07	50.57	457.30	53.18	505.13	44.43	518.27	26.36	546.20	66.45	549.07	24.64
U	127.53	2.75	244.47	3.86	357.10	5.05	467.33	5.70	578.27	3.14	683.10	3.12	790.27	3.67	890.70	2.77
UCB	142.00	0.00	592.30	0.11	773.30	7.30	800.90	0.72	965.00	4.06	978.90	0.47	987.80	0.23	995.23	0.16
ARW	241.13	1.69	471.43	3.94	581.83	3.56	660.60	6.13	727.73	4.55	783.00	4.35	838.90	3.82	886.57	2.85
AP	226.53	2.71	332.80	3.94	427.90	5.10	522.23	7.28	614.30	4.73	700.80	4.21	792.27	5.65	875.07	2.62
MAB	142.00	0.00	592.27	0.14	770.03	10.92	800.60	1.03	964.67	3.23	978.50	0.36	987.60	0.21	995.13	0.17
wsiz = 4																
N ₁	1		2		3		4		5		6		7		8	
OR	198.53	0.40	608.00	1.58	807.13	0.93	905.95	0.81	1000.00	0.00	1000.00	0.00	1000.00	0.00	1000.00	0.00
IM	85.17	3.08	509.30	1.30	735.02	1.52	762.58	1.62	965.50	3.59	969.95	2.83	973.15	1.94	976.92	2.29
GR	2.50	0.00	2.50	0.00	2.50	0.00	2.50	0.00	2.50	0.00	2.50	0.00	2.50	0.00	2.50	0.00
EGR	58.88	3.30	414.82	47.13	450.02	38.76	486.55	30.67	517.70	25.94	533.13	37.05	570.33	46.05	589.15	42.27
U	128.97	3.40	243.75	3.76	356.65	3.73	468.70	6.32	580.48	4.35	682.55	5.40	788.05	3.53	887.00	1.70
UCB	164.50	0.00	566.00	1.70	805.83	0.65	835.58	0.17	854.70	1.39	974.70	2.21	987.00	0.26	996.17	0.17
ARW	241.32	2.49	471.05	3.49	581.15	4.74	659.90	2.20	727.42	4.47	780.85	3.21	836.15	2.04	885.52	3.15
AP	225.35	1.69	333.38	4.67	428.32	4.01	522.75	4.28	614.13	3.41	700.67	4.47	792.88	2.04	875.10	2.23
MAB	164.50	0.00	565.60	1.55	806.20	0.42	835.60	0.17	854.42	1.12	976.25	0.93	987.00	0.26	996.17	0.12

Table 2: Results for $wsiz \in \{1 \dots 4\}$

Table 3: Results for $wsiz e \in \{5 \dots 8\}$

wsiz e = 5																
N_1	1		2		3		4		5		6		7		8	
OR	166.70	0.19	601.20	0.00	800.60	0.00	872.56	0.75	937.56	0.61	1000.00	0.00	1000.00	0.00	1000.00	0.00
IM	91.52	4.66	582.40	1.23	768.56	2.62	794.54	2.41	819.16	2.70	970.48	2.63	971.58	3.07	975.16	2.57
GR	3.00	0.00	3.00	0.00	3.00	0.00	3.00	0.00	3.00	0.00	3.00	0.00	3.00	0.00	3.00	0.00
EGR	58.20	2.15	430.34	73.93	484.98	51.39	536.28	19.68	528.46	23.46	572.24	31.44	599.74	10.36	611.76	21.17
U	130.52	2.92	243.38	3.67	358.28	6.14	466.40	7.12	576.94	4.82	683.56	7.01	788.42	3.33	887.84	1.35
UCB	184.80	0.00	574.94	1.93	738.44	0.86	854.08	0.94	872.42	0.24	887.38	0.50	974.80	7.01	994.22	0.37
ARW	240.68	1.75	469.22	3.33	580.20	3.40	659.10	4.04	724.72	2.62	781.58	3.47	834.36	3.61	883.46	2.07
AP	226.30	2.69	334.38	4.45	430.52	4.35	526.74	6.74	616.26	7.43	707.02	5.61	794.02	3.51	875.70	1.91
MAB	184.80	0.00	574.26	0.30	738.52	0.81	854.38	0.81	872.44	0.26	887.20	0.45	980.04	4.16	994.14	0.28
wsiz e = 6																
N_1	1		2		3		4		5		6		7		8	
OR	144.12	0.27	578.72	1.33	768.27	0.65	862.33	0.66	909.57	0.74	955.85	0.47	1000.00	0.00	1000.00	0.00
IM	96.57	3.11	510.23	0.49	671.08	0.99	813.60	2.15	835.00	1.58	857.20	3.18	973.15	3.86	974.35	2.52
GR	3.50	0.00	3.50	0.00	3.50	0.00	3.50	0.00	3.50	0.00	3.50	0.00	3.50	0.00	3.50	0.00
EGR	59.37	2.32	433.13	28.65	498.98	34.03	510.15	43.94	568.27	53.50	604.88	23.08	647.55	29.97	662.83	26.99
U	128.28	3.79	244.98	4.94	357.35	4.60	467.82	4.30	575.32	4.74	684.33	4.22	787.58	3.63	885.80	1.38
UCB	201.17	0.00	561.00	0.00	760.25	0.35	840.15	7.30	880.75	0.84	898.58	0.23	910.23	0.27	974.58	8.64
ARW	240.33	2.13	468.58	2.82	579.22	1.84	659.07	3.39	722.50	2.74	781.13	3.66	834.12	3.63	883.23	1.69
AP	224.98	2.37	335.73	5.57	435.22	6.26	527.13	5.30	615.90	5.86	708.10	4.19	796.27	2.89	877.12	2.90
MAB	201.17	0.00	561.00	0.00	759.60	1.41	837.42	6.16	881.77	2.46	898.58	0.21	910.10	0.34	975.27	14.07
wsiz e = 7																
N_1	1		2		3		4		5		6		7		8	
OR	127.13	0.16	573.14	0.00	756.39	0.98	857.71	0.00	897.54	0.70	932.19	0.28	967.11	0.38	1000.00	0.00
IM	100.46	3.29	556.69	1.81	702.44	1.34	825.40	4.26	847.14	1.61	865.43	1.70	886.01	1.50	976.46	3.01
GR	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00
EGR	59.20	2.05	470.93	34.59	519.37	30.66	542.99	22.51	607.64	38.34	645.67	31.45	675.43	27.44	691.09	18.53
U	129.63	2.80	245.46	5.64	355.79	5.33	464.36	4.79	577.07	8.15	682.26	4.34	785.00	3.19	884.43	1.14
UCB	217.14	0.00	548.91	0.78	711.27	1.30	806.94	0.14	863.39	0.07	905.13	0.85	917.91	0.12	929.51	0.19
ARW	240.41	2.14	467.09	2.75	582.86	6.93	656.40	3.26	721.04	2.26	781.59	4.08	833.70	1.99	882.81	1.36
AP	224.63	1.59	335.39	3.13	434.30	5.25	527.60	4.27	620.79	3.49	707.79	3.87	795.66	5.08	877.26	2.09
MAB	217.14	0.00	548.80	0.69	710.41	1.12	806.60	0.57	863.49	0.30	905.33	1.22	917.93	0.29	929.46	0.11
wsiz e = 8																
N_1	1		2		3		4		5		6		7		8	
OR	113.85	0.13	563.27	1.10	751.13	0.00	837.64	0.61	893.13	0.99	920.81	0.47	947.70	0.59	974.55	0.37
IM	100.39	4.08	511.00	1.02	723.50	2.75	759.10	1.79	853.79	2.95	871.41	1.30	888.58	1.72	905.36	1.06
GR	4.50	0.00	4.50	0.00	4.50	0.00	4.50	0.00	4.50	0.00	4.50	0.00	4.50	0.00	4.50	0.00
EGR	60.30	2.98	480.61	21.00	527.46	32.21	592.42	21.79	622.77	31.81	663.14	37.08	690.80	28.90	708.16	11.92
U	129.57	3.12	242.61	4.62	356.14	3.26	468.79	7.13	579.90	4.52	681.92	3.25	786.23	4.28	884.59	1.02
UCB	229.13	0.00	539.89	0.48	734.38	3.73	790.52	1.06	848.60	0.13	878.41	3.08	923.50	0.75	934.95	0.26
ARW	240.40	2.61	468.75	2.44	579.44	2.06	659.15	2.24	726.02	4.13	780.99	2.70	833.80	1.85	882.02	1.63
AP	224.30	1.48	337.06	4.74	433.56	3.32	529.66	2.60	621.54	2.91	707.58	4.07	797.25	3.75	876.99	1.37
MAB	229.13	0.00	539.80	0.44	733.01	4.73	789.67	0.43	848.65	0.21	878.75	3.28	923.66	0.51	934.94	0.21

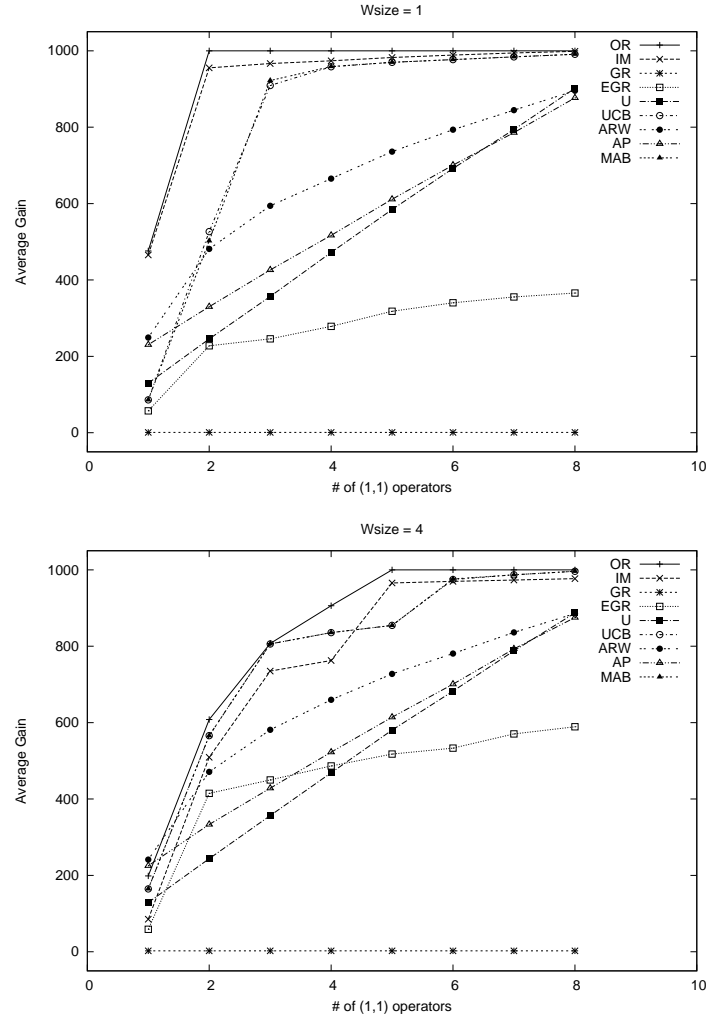


Figure 1: Experiments with different windows size

5.3. Comments

- Considering the first column of tables 2 and 3, i.e. scenario with $N_{op} = 1$, it should be noted that the suboptimal optimization solution presented in Section 4.4 provides better results than *OR*. This is due to the fact that *OR* is a myopic oracle while the suboptimal circular policy based on circular scenarios takes into account sequences of applications (in a dynamic programming fashion).
- As expected, greedy strategies cannot insure a good schedule of the different operators in this non stationary context.
- *UCB* and *MAB* are equivalent since no restart of the learning process is used here — all *MAB* parameters are set to 0. Note that these parameters are useful when the distribution of gains is varying according to epochs (epochs based scenarios previously described). Here, since gains change continuously, such restart strategy is not efficient — or would induce too much noise in the learning process. This has been checked by experiments. Note that here the standard UCB can be used directly without any reward normalization factor, since all rewards range from 0 to 1.
- According to the results mentioned in Section 4.4, the uniform choice *U* provides the same results independently from the size of the window.
- ARW and AP obtain comparable results, with a slight superiority of ARW. Both policies give more importance to the best operator along the search. Such a strategy is not necessary well suited for this problem, but is rather efficient when N_1 is low. The balance adjustment between \mathbb{O} and \mathbb{I} operators is insured by the p_{\min} parameter. Note that these policies obtain similar results independently from the size of the window, which seems to mean that their behaviour is close to a uniform choice, but restricted to the most efficient operators. When N_1 increases, they become indeed equivalent to *U*. We have checked that, studying the sequence of selected operators, no clear repeated sequence of operators can be observed ; however the total amount of \mathbb{I} operators used remains similar in the different problem configurations.
- Concerning UCB, MAB and IM, we may distinguish several different ranges of results:

- When $N_1 = 1$, UCB (and MAB, but in the following, we will only mention UCB) and IM cannot insure a good management of the balance between \mathbb{O} and \mathbb{I} operators, certainly due to their exploration component — ie. noise in IM and right member in UCB formula (see section 3.1.2). .
 - When $1 \leq N_1 \leq wsize$, UCB seems to provide better results than IM on most instances. While N_1 increases, the gap between the policies reduces. Moreover, the policies are more and more close to the oracle.
 - When $N_1 = wsize + 1$, one observes that OR is able to compute an optimal schedule (1000). Indeed, it is possible here to alternate between \mathbb{I} operators only. We also observe that IM is also able to increase its performance, and becomes then better than UCB.
 - When $N_1 > wsize + 1$, the problem becomes easier since a uniform choice \mathbb{U} is then a reasonably efficient policy. Here, IM and UCB have a performance close to OR.
- We may remark that the behaviour of the different policies is almost the same when dimensions of the problem increase, shifted according to the values of N_1 and $wsize$. We have conducted experiments on other dimensions (see Figure 2) with similar observations.

6. Conclusion

In this paper, we have proposed a new model for simulating non stationary operators in search algorithms that should alternate between intensification and diversification stages in their search processes. The abstract model that is defined here may serve to evaluate the performance of operator selection policies in these search algorithms. We proposed here an experimental studies of different classic operator selection policies in order to highlight their respective advantages and drawbacks in such search scenarios.

Our model can be considered as possible a surrogate model in order to design new adaptive search algorithms that aim at solving general optimization problems without focusing on specific dedicated operators or search heuristics. For a reinforcement learning point of view, our model corresponds to specific restless bandit problems that could be used to model different real applications as soon as the efficiency of a given action decreases according to successive frequent uses.

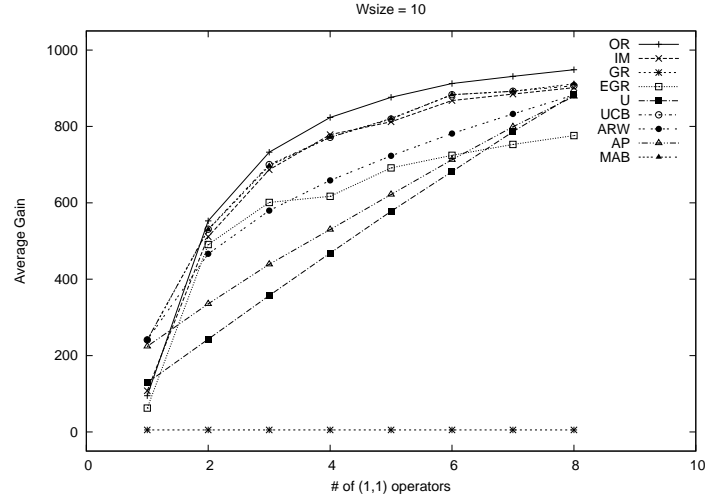


Figure 2: Experiments with $wsiz e = 10$

References

- Auer, P., 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3, 397–422.
- Auer, P., Cesa-Bianchi, N., Fischer, P., 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47 (2-3), 235–256.
- Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K., 2002. A racing algorithm for configuring metaheuristics. In: *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 11–18.
- Bradt, R. N., Johnson, S. M., Karlin, S., 1956. On sequential designs for maximizing the sum of n observations. *The Annals of Mathematical Statistics* 27 (4), 1060–1074.
- Candan, C., Goëffon, A., Lardeux, F., Saubion, F., 2012. A dynamic island model for adaptive operator selection. In: *Genetic and Evolutionary Computation Conference (GECCO'12)*. pp. 1253–1260.
- Costa, L. D., Fialho, Á., Schoenauer, M., Sebag, M., 2008. Adaptive operator selection with dynamic multi-armed bandits. In: *Genetic and Evolutionary*

- Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008. ACM, pp. 913–920.
- Eiben, A. E., Michalewicz, Z., Schoenauer, M., Smith, J. E., 2007. Parameter control in evolutionary algorithms. In: *Parameter Setting in Evolutionary Algorithms*. pp. 19–46.
- Feldman, D., 1962. Contributions to the "two-armed bandit" problem. *The Annals of Mathematical Statistics* 33 (3), 847–856.
- Fialho, Á., Dec. 2010. Adaptive operator selection for optimization. Ph.D. thesis, Université Paris-Sud 11, Orsay, France.
- Fialho, Á., Costa, L. D., Schoenauer, M., Sebag, M., 2008. Extreme value based adaptive operator selection. In: *Parallel Problem Solving from Nature - PPSN X, 10th International Conference, Proceedings*. Vol. 5199 of *Lecture Notes in Computer Science*. Springer, pp. 175–184.
- Fialho, Á., Costa, L. D., Schoenauer, M., Sebag, M., 2010. Analyzing bandit-based adaptive operator selection mechanisms. *Ann. Math. Artif. Intell.* 60 (1-2), 25–64.
- Gittins, J. C., 1979. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)* 41 (2), 148–177.
- Hamadi, Y., Monfroy, E., Saubion, F., 2012. *Autonomous search*. Springer-Verlag.
- Hinkley, D. V., 1970. Inference about the change-point in a sequence of random variables. *Biometrika* 57 (1), 1–17.
- Hoos, H. H., 2012. *Autonomous Search*. Springer Verlag, Ch. Automated Algorithm Configuration and Parameter Tuning, pp. 37–71.
- Hutter, F., Hoos, H., Leyton-Brown, K., Stützle, T., Sep. 2009. ParamILS: an automatic algorithm configuration framework. *J. Artif. Int. Res.* 36 (1), 267–306.
- Koulouriotis, D., Xanthopoulos, A., 2008. Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. *Applied Mathematics and Computation* 196 (2), 913 – 922.

- Lobo, F., Lima, C., Michalewicz, Z. (Eds.), 2007. Parameter Setting in Evolutionary Algorithms. Vol. 54 of Studies in Computational Intelligence. Springer.
- Mahajan, A., Teneketzis, D., 2008. Multi-armed bandit problems. In: Hero, AlfredO., I., Castan, D., Cochran, D., Kastella, K. (Eds.), Foundations and Applications of Sensor Management. Springer US, pp. 121–151.
- Maturana, J., Fialho, A., Saubion, F., Schoenauer, M., Sebag, M., 2009. Compass and dynamic multi-armed bandits for adaptive operator selection. In: Proc. of IEEE Congress on Evolutionary Computation CEC. IEEE, pp. 365–372.
- Nannen, V., Smit, S. K., Eiben, Á. E., 2008. Costs and benefits of tuning parameters of evolutionary algorithms. In: G. Rudolph et al. (Ed.), Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, September 13-17, 2008, Proceedings. Springer, pp. 528–538.
- NICTA, 2014. Minizinc.
URL <http://www.minizinc.org/>
- Robbins, H., 1952. Some aspects of the sequential desing of experiments. Bulletin Amrican Mathematical Society (55), 527–535.
- Rodman, L., 1978. On the many-armed bandit problem. The Annals of Probability 6 (3), 491–498.
- Scilab Enterprise, 2014. Scilab.
URL www.scilab.org
- Smit, S., G.Eiben, 2009. Comparing parameter tuning methods for evolutionary algorithms. In: Proceedings of the IEEE Congress on Evolutionary Computation.
- Sutton, R., Barto, A., 1998. Reinforcement Learning: An Introduction. MIT Press.
- Thierens, D., 2005. An adaptive pursuit strategy for allocating operator probabilities. In: Genetic and Evolutionary Computation Conference, GECCO. ACM, pp. 1539–1546.

- Thierens, D., 2007. Adaptive Strategies for Operator Allocation. In: Lobo, F., Lima, C., Michalewicz, Z. (Eds.), *Parameter Setting in Evolutionary Algorithms*. Springer Verlag, pp. 77–90.
- Whittle, P., 1988. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability* 25, 287–298.
- Wolpert, D., Macready, W., 1997. No free lunch theorems for optimization. *IEEE Trans. Evolutionary Computation* 1 (1), 67–82.